6И

УДК 004.89+004.04

С.А. Ильин
Международный научно-учебный центр информационных технологий и систем
НАН Украины и МОН Украины, г. Киев, Украина
iljin.sergiy@gmail.com

О проблемах интеграции интеллектуальных технологий в системах управления мобильных роботов

Рассмотрена специфика создания систем управления мобильных роботов с использованием готовых программных компонентов, реализованных сторонними разработчиками. Показана актуальность создания эффективных внутренних структур взаимодействия таких компонентов и родственность данной задачи мультиагентным подходам. Приведен пример реализации системы управления с трехуровневой иерархией функциональных компонентов как интеллектуальных агентов.

Введение

В последние годы значительно вырос интерес к построению систем искусственного интеллекта в так называемом мультиагентном подходе [1-3]. В робототехнике данная парадигма используется, как правило, только для реализации в разных системах управления (СУ) таких функций, как обучение, целенаправленное функционирование или принятие решений. При этом недостаточно внимания уделяется организации эффективной структуры информационного взаимодействия составляющих компонентов СУ, что является достаточно важным аспектом их работы. Современные робототехнические системы должны объединять в себе целый набор сложных аппаратных средств и алгоритмов статистической обработки и анализа мультимодальных данных, которые включают в себя несколько принципиально разных уровней обработки и абстракции информации [4] и организовывают параллельно обработку всех внутренних информационных потоков [5]. Кроме того, окончательный продукт должен иметь эффективную, гибкую, легко масштабируемую архитектуру.

Построение такой СУ требует решения целого ряда задач, многими из которых занимаются отдельные исследовательские лаборатории или целые институты. Как результат, существует множество различных относительно узко специализированных программных разработок, в которых достаточно успешно решены задачи навигации, построения карт окружающей среды, мониторинга территорий и т.п. Часть подобного рода разработок изначально ориентированны на повторное использование как базовой платформы для будущих СУ или просто готовы к использованию в виде программных компонентов. В такой ситуации возник ряд программных продуктов – наборов инструментальных средств, которые пытаются интегрировать достижения разных разработчиков [6-9]. Главной задачей данных продуктов является обеспечение необходимым инструментарием: интерфейсами, средствами обмена информацией и т.п. для интеграции уже готовых программных решений в единое целое, что значительно упрощает процесс создания программного обеспечения СУ и позволяет концентрироваться на какой-либо своей конкретной задаче, имея при этом возможность тестировать ее на реальном роботе в условиях реальной среды функционирования, что, несомненно, очень важно на стадии разработки.

Компонентно-ориентированная разработка

Разработка приложений, изначально собираемых из уже готовых составляющих, имеет некоторые особенности. Не вдаваясь в известные детали объектно-ориентированного программирования (ООП) [10], можно утверждать, что его основная идея состоит в том, чтобы привести процесс разработки программного обеспечения (ПО) в соответствие с ментальной моделью восприятия человеком реальных или виртуальных объектов как единиц ПО. Таким образом, сделав этот процесс более простым для понимания, ООП и близлежащие дисциплины объектно-ориентированного дизайна и объектно-ориентированного анализа фокусируются на моделировании взаимодействий сущностей реального мира, пытаясь создать «глаголы» и «существительные», которыми можно пользоваться интуитивно, как программисту, так и конечному пользователю, вплоть до того, что последний может вносить нужные незначительные изменения в ПО.

Программные компоненты, в отличие от объектов, не стремятся к интуитивности восприятия. Компонентно-ориентированная разработка исходит из предположения, что ПО должно состоять из ранее разработанных компонентов, объединенных функциональными связями [11]. Она допускает, что определение полезности компонентов, в отличие от объектов, может быть далеким от интуитивности. При этом упускаются антропоморфизм, прозрачность функциональности и оптимизм по поводу возможности создавать конечным пользователем новую функциональность ПО.

В некотором роде здесь можно говорить о новой парадигме программирования: *компонентно-ориентированное программирование*.

Взаимодействие компонентов в СУ МР и мультиагентность

Модель взаимодействия программных компонентов СУ МР в качестве независимых составляющих можно определить как мультиагентную. Здесь под термином агент подразумевается программный или аппаратный модуль, блоки решения задач которого размещены в некой среде, и каждый из них способен к гибким, автономным и организованным действиям, направленным (или же нет) на достижение поставленной цели [12]. В данной формулировке чётко прослеживается родственность архитектуры взаимодействия, распределения классов задач и основных специфических свойств компонентов СУ МР с агентами мультиагентных систем:

– *ситуативность* – любой агент находится в определенной для него среде и может активно влиять на эту среду: запускать дополнительные сервисы, формировать новые задачи для неё и т.п.;

– *автономность* – каждый из агентов не втянут в общий ход выполнения текущего задания МР, но имеет свои конкретные цели и сам принимает решение, какие действия проводить для их достижения. То есть каждый агент может существовать как самодостаточный сервис, но вместе с тем он должен быть способным предоставлять часть своей функциональности окружению;

– *гибкость* – агент может получить внешние указания-стимулы для работы и должным образом реагировать на них;

– *супервизорная активность* – агент проводит не только деятельность, направленную на достижение своих персональных целей, а и контроль, диагностику и сигнализацию своевременного исполнения поставленных им целей другим агентам;

– *социальность* – агенты взаимодействуют между собой при помощи организации обмена запросами на обработку информации, служебными, сообщениями, реализуемыми системными событиями и интерфейсами, так как это необходимо для достижения поставленной цели.

В некотором смысле данное определение агента совпадает с уже устоявшимся представлением о такого же рода сущностях, сложившимся в современных операционных системах вычислительной техники как разновидности управляющих систем. Единственное отличие здесь состоит в среде функционирования агента, но это не столь существенно, так как она не может быть гомогенной в СУ МР. Поэтому отдельные программные процессы, потоки [13] и аппаратные модули управления периферийным оборудованием обобщенно могут описываться мультиагентой моделью взаимодействия.

Компонентами СУ МР могут являться программные реализации, например, следующих интеллектуальных технологий: визуального распознавания, глобального/локального позиционирования в пространстве, построения карт среды функционирования, планирования действий и т.п. Каждый из таких компонентов ориентирован на решение своих узкоспециализированных, но весьма сложных задач, а объединяющей средой для них является вычислительная среда, в которой работает СУ МР, интерфейсы ее составляющих и соответствующего периферийного оборудования.

Объекты и агенты СУ МР

В процессе усложнения реализуемых на практике СУ МР различия между агентами и объектами – конструктивами ООП – становятся менее четкими. Для начала рассмотрим исторически традиционный объектно-ориентированный подход. Сам по себе объект не производит никакой активности, все его действия вызываются внешними служебными запросами. Как следствие, объект не имеет характерного для компонентов СУ МР свойства автономности и не может принимать решения о том, что делать в специфической ситуации. Более того, в традиционных объектных приложениях невозможно полноценно определить «среду»: объекты инкапсулируют ресурсы только в терминах их внутренних атрибутов или ощущают внешний мир только в терминах имен/ссылок на другие объекты.

С другой стороны, объекты в ПО современных СУ МР в каком-то смысле используются в новом для них представлении, более близком к данному выше определению агентов, как минимум с точки зрения супервизорной активности. Объекты могут быть активными и могут включать внутренние вычислительные потоки, дающие им возможность выполнять специфические задачи. Они также могут обслуживать внешние запросы и должны защищать себя от обработки ошибочных или контекстно недопустимых на данном этапе действий. Другими словами, они могут принимать решение о том, выполнять или не выполнять какое-либо действие. Наконец, сосуществование в СУ МР обоих видов активных и пассивных объектов обеспечивает четкое распределение между активной вычислительной ее частью и окружающими внешними ресурсами (средой).

Итак, в ходе функционирования объекты должны получать доступ к внешним ресурсам и должны часто иметь дело с неопределенными ситуациями (обработка исключительных ситуаций или событий). Это может требовать полного моделирования ситуативности поведения (такого, как четкая идентификация контекстных зависимостей между компонентами СУ).

Когда объекты с подобными свойствами используются при функционировании СУ МР, их наблюдаемое поведение, с точки зрения разработчика, такое же, как у автономного агента.

В дополнение к тому, что современные объекты и компоненты СУ МР на практике обретают характеристики, делающие их активность такой же, как у агентов, которые сами по себе часто реализуются в виде активных составляющих такого рода программных комплексов (потоки выполнения, обработка исключений и событий). Поэтому вышеска-

занное следует понимать в таком значении: не агенты сами по себе добавляют эффективность функциональности СУ МР, а только их грамотное использование в строго продуманных внутренних структурах взаимодействия. Более того, даже с точки зрения технологий программирования, агентно-ориентированная абстракция очень хорошо подходит к разработке сложных программных систем, чем, несомненно, являются современные СУ МР.

Структура СУ МР с использованием компонентов разных разработчиков

Создание СУ МР на базе компонентов, полученных от разных разработчиков, имеет ряд сложностей. Основным из них является отсутствие строгой структуры информационного обмена между отдельными компонентами, что в будущем может породить множество различных проблем, таких как: значительные объемы обмена информацией между различными компонентами, конкуренция за ресурсы вычислительной среды, потеря гибкости архитектуры в целом (рис. 1).

Конечно же, подход, состоящий в интеграции готовых компонентов сторонних производителей, дает достаточно быстрый результат. И если главная задача конкретной разработки – тестирование в реальной среде каких-либо интеллектуальных алгоритмов, экспериментального периферийного оборудования и т.п., то вышеупомянутыми проблемами можно пренебречь. Если же задача состоит непосредственно в разработке эффективной СУ МР, позволяющей организовывать целеустремленное функционирование робототехнического комплекса с перспективами дальнейшего усовершенствования, поддержкой различных конфигураций периферийного оборудования и т.п., то любые потенциальные структурные проблемы нельзя упускать из виду. В противном случае, их будущие проявления приведут к значительному усложнению разработки и сопровождения комплекса в целом, что крайне нежелательно.

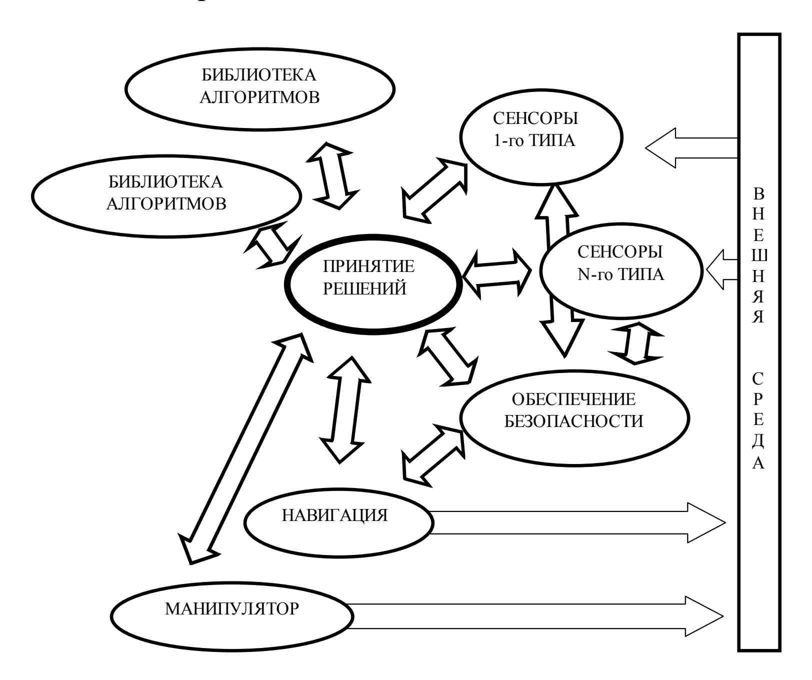


Рисунок 1 – Архитектура СУ МР, полученная при интеграции различных компонентов сторонних производителей

Синхронизация во времени данных, получаемых от различных компонентов СУ

Подавляющее большинство существующих компонентов СУ МР имеют дискретную среду функционирования. Это продиктовано существующими вычислительными платформами, измерительным оборудованием и т.п. Так как СУ МР должна стремится к функционированию в режиме, приближенном к реальному времени, а существующие аппаратные средства не позволяют проводить весь необходимый комплекс вычислений параллельно, возникает задача синхронизации, или сведения полученных данных к единому масштабу. Например, если данные, полученные от агента, следящего за положением МР в пространстве, не будут полностью синхронны с данными, полученными от агента, следящего за наличием препятствий, то восстановить реальное положение последних в пространстве представляется невозможным. Это влечет за собой принятие ложных решений в навигации и т.п. Подобных ситуаций можно привести достаточно много.

Частично явление рассинхронизации можно компенсировать введением временных меток во все структуры данных, что позволит использовать математическое моделирование для так называемой виртуальной синхронизации данных, которая состоит в следующем: если интервал времени между несинхронными данными не превышает допустимого уровня, то более устаревшее значение можно вычислить как

$$x_i' = x_i + dx_i = x_i + \frac{\partial f_i(t)}{\partial t} dt_i', \tag{1}$$

где x_i' – вычисляемое значение, dx_i – ожидаемый прирост, $f_i(t)$ – закон изменения во времени величины x_i. Так как со временем накапливается ошибка метода моделирования $\Delta f_i \sim t$, вычисляемая величина должна уточняться непосредственным измерением, ошибка не зависит от времени. Для небольших dt_i между отдельными измерениями ошибка метода моделирования Δf_i соотносится с ошибкой рассинхронизации во времени Δx_i следующим образом:

$$\Delta f_i << \Delta x_i, \tag{2}$$

поэтому можно пользоваться (1) без существенной утраты точности.

Всегда следует учитывать тот факт, что далеко не все моды входящей информации поддаются моделированию. Поэтому способности применять этот подход на практике, определять его ошибки и границы использования должны быть строго инкапсулированы внутри соответствующего компонента СУ МР, для того чтобы не порождать дополнительных контекстных зависимостей и сделать этот процесс максимально незаметным для окружения. Другими словами, компонент всегда должен стремиться предоставлять максимально приближенные к текущему моменту времени данные.

Эффективность структуры компонентно-ориентированной СУ МР

Предлагаемый автором подход к разработке СУ МР основывается на том, что основные функциональные группы компонентов отвечают за функционирование и взаимодействие аппаратных средств, реализацию простейших базовых алгоритмов поведения и принятия решений о целенаправленных действиях. В соответствии с этим, они объединяются в иерархическую структуру из элементов 3 уровней:

1) команды – неделимые элементы, отвечающие за непосредственное общение с аппаратным обеспечением, объектами вычислительной среды, алгоритмами обработки информации. Каждая команда или библиотека однотипных команд инкапсулируется в отдельном компоненте, отвечающем за ее выполнение, диагностику, логическое сопровождение;

2) поведения – алгоритмы, реализующие типичные последовательности команд, связанные операторами условного перехода;

3) задания – алгоритмы, реализующие последовательности актов поведений или принятия решений, связанных операторами условного перехода.

В приведенной иерархии разбиение СУ МР на составляющие компоненты является важнейшей задачей, так как именно это и определяет гибкость полученной системы в использовании, возможность применения одних и тех же компонентов в ходе построения различных алгоритмов функционирования. Особое место здесь занимает построение уровня команд. Для того чтобы разные компоненты верхних уровней могли их использовать, все команды должны иметь одинаковые интерфейсы (или отдельные группы интерфейсов), поскольку командой может быть как запрос на программно реализованный процесс распознавания объектов, присутствующих на текущем изображении бортовой видеокамеры, так и запрос к интерфейсу управления двигателями колес для выполнения движения МР вперед. Это можно реализовать только при условии обеспечения высокого уровня абстракции отдельно взятых команд от их конкретных реализаций, типов обрабатываемой информации и т.п.

Также важным является разработка эффективного протокола взаимодействия между компонентами СУ МР, который должен минимизировать количество производимых транзакций.

Одной из основных задач функционирования СУ МР должна быть самодиагностика и поддержание работоспособности системы в целом. Это реализуется так называемым специальным заданием, которое выполняется всегда при активном состоянии МР. В его обязанности входят: инициализация всего периферийного оборудования, слежение за важными параметрами функционирования, разрешение конфликтов доступа к оборудованию разных компонентов и т.п. Данное задание можно определить как «базовый процесс», описанный в [14].

В сложных СУ МР на «базовый процесс» возлагается еще одна очень важная функция. Выполнение некоторых заданий МР требует достаточно длительного промежутка времени. В таких случаях может потребоваться их остановка для выполнения другого задания – например, подзарядки аккумуляторов, с последующим возобновлением выполнения предыдущих действий. Именно «базовый процесс» инициирует данное действие, запускает на выполнение необходимое промежуточное задание, отслеживает его завершение и восстанавливает предыдущее, выполняя, таким образом, функции диспетчера целенаправленной деятельности МР в целом.

Выводы

Как сказано выше, сложность СУ МР пока не позволяет сконцентрировать ее разработку полностью в одних руках. Подход, базирующийся на интегрировании уже существующих компонентов, позволяет значительно ускорить разработку и достаточно легко получить полноценную действующую систему, а также дает воз-

можность повторного использования ранее созданных компонентов или дальнейшей интеграции в сторонние СУ. Однако такие системы могут иметь успех только при условии достаточного внимания к созданию эффективной внутренней структуры информационного взаимодействия составляющих компонентов как интеллектуальных агентов.

Литература

1. Wooldridge M.J., Jennings N.R. Intelligent agents: Theory and practice // The Knowledge Engineering Review. – 1995. – № 10 (2). – P. 115-152.
2. Developing Multiagent Systems: the Gaia Methodology / F. Zambonelli, N.R. Jennings, M.J. Wooldridge // ACM Transactions on Software Engineering and Methodology. – 2003. – №. 12 (3). – P. 317-370.
3. Zambonelli F., Omicini A. Challenges and Research Directions in Agent-Oriented Software Engineering // Journal of Autonomous Agents and Multiagent Systems. – 2004. – № 9 (3). – P. 253-283.
4. Васильев И.А, Ляшин А.М. Разработка программного обеспечения системы управления робототехническими средствами // Тр. школы-семинара «Адаптивные роботы – 2004», (8-10 июня 2004 г.) С.-Петербург. – С. 55-58.
5. Ильин С.А. Особенности реализации программного обеспечения системы управления мобильного робота // Управляющие системы и машины. – 2007. – № 4. – С. 28-42.
6. MARIE. – Режим доступа: http://marie.sourceforge.net/.
7. Carnegie Mellon Robot Navigation Toolkit. – Режим доступа: http://carmen.sourceforge.net/.
8. RobotFlow – Open Source Robotics Toolkit. – Режим доступа: http://robotflow.sourceforge.net/.
9. Microsoft Robotics. – Режим доступа: http://msdn.microsoft.com/en-us/robotics/default.aspx.
10. Booch G. Object-oriented Analysis and Design (second edition). Addison Wesley, Reading (MA), 1994.
11. Nierstrasz O., Gibbs S., Tsichritzis D., Component-Oriented Software Development. – Режим доступа: http://www.iam.unibe.ch/~scg/Archive/OSG/Nier92bCOSD.pdf.
12. Рассел С., Норвиг П. Искусственный интеллект: современный подход. – 2-е изд. – М.: Издательский дом «Вильямс», 2006. – 1408 с.
13. Тененбаум Э. Современные операционные системы. – СПб.: Питер, 2002. – 1040 с.
14. Рапопорт Г.Н., Герц А.Г. Искусственный и биологический интеллекты. – М.: КомКнига, 2005. – 312 с.

С.О. Ільїн

Про проблеми інтеграції інтелектуальних технологій в системах керування мобільних роботів

Розглянуто специфіку створення систем керування мобільних роботів з використанням готових програмних компонентів, реалізованих сторонніми розробниками. Показано актуальність створення ефективних внутрішніх структур взаємодії таких компонентів та спорідненість даної задачі мультиагентним підходам. Наведено приклад реалізації системи керування із трирівневою ієрархією функціональних компонентів як інтелектуальних агентів.

S.O. Ilyin

Some Problems of Integration of Artificial Technologies into Control System of the Mobile Robot

This paper deals to the multiagent approach to the mobile robot's control system development with using of previously known components of the third party developers. It is argued high importance of optimal informational interaction between components and connection of such task to multiagent approach. An example of the robot control's system three-levels components hierarchy as an intelligent agents is given.

Статья поступила в редакцию 10.07.2008.